# TEMPORAL RELATIONS IN TEXTS

Mohamed Suliman `Student No:17327728`

## Abstract

This report addresses temporal reasoning associated with news texts. Drawing on the TimeML annotations in the TimeBank corpus, and observations about the likelihood of the 13 Allen relations on intervals of time in the absence of conditioning context, this project explores the string represenations of texts given by the approach of finite state temporality derived from a TimeML annotated corpus. Given the sequence of events annotated in texts, the report analyses the sequences of Allen relations that hold among events. Knowing likely sequences of Allen relations supports default reasoning about pairwise event relations.

## 1 INTRODUCTION

Considerable work has gone into how one should go about representing temporal information in natural language. The work presented in this report is rooted deeply in the approach pioneered in Allen (1983) [1]. When we think about events occuring during a specified interval of time *e.g a telephone ringing*, Allen shows that there exist only 13 possible relations between two intervals of time during which two separate events are occuring, no matter how long they may be. These are defined by the following six relations: before, during, overlaps, meets, starts, finishes, as well as their inverses. The thirteenth relation is the case when both intervals are equal. Representing these relations in an intuitive and computationally efficient manner is done in Woods et. al. [6], via the use of finite temporatily strings.

Taking the set $A$ as the finite set of temporal propositions, *fluents*, or event names, we have a string $s = \alpha_1 ... \alpha_n$ of subsets $\alpha_i$ of $A$, with it being understood that at position $i, 1 \leq i \leq n$, every fluent $a \in \alpha_i$ holds true. The strings are to be read from left to right providing a nice timeline view of events. Take the example where we have $A = \{tr, je\}$, with $tr$ = *"the telephone ringing"* and $je$ = *"john enters the room"* . Then if we know that the telephone rang before John entered the room, this information may be represented by the string $s = \{\}\{tr\}\{\}\{je\}\{\}$. In order to not confuse set notation with our strings, we will henceforth refer to strings with a pipe notation, similar to the boxes used in [6]. So the above string would become `| |tr| |je| |`, showing that the telephone rang before John entered. The empty box, `| |`, represents the empty set $\varnothing$, the interval in time where none of the events $a \in A$ are occuring. Table 1 shows all the possible ways in which these two events could have played out in time. Note that each string is bounded with an empty box on both sides so we do not have events that occur for an infinite amount of time *i.e* each event interval has a defined start and end point.

Table 1: The possible ways that the telephone ringing and john entering the room could have played out represented by the 13 Allen relations.

| Relation | String |
|---|---|
| equal (e) | \| \| tr, je \| \| |
| before (b) | \| \| tr \| \| je \| \| |
| after (bi) | \| \| je \| \| tr \| \| |
| during (d) | \| \| je \| tr, je \| je \| \| |
| contains (di) | \| \| tr \| je, tr \| tr \| \| |
| overlaps (o) | \| \| tr \| tr, je \| je \| \| |
| overlapped-by (oi) | \| \| je \| tr, je \| tr \| \| |
| meets (m) | \| \| tr \| je \| \| |
| met-by (mi) | \| \| je \| tr \| \| |
| starts (s) | \| \| tr, je \| je \| \| |
| started-by (si) | \| \| tr, je \| tr \| \| |
| finishes (f) | \| \| je \| tr, je \| \| |
| finished-by (fi) | \| \| tr \| tr, je \| \| |

TimeBank is a corpus of 183 news articles manually annotated according to the TimeML standard of annotating temporal information in texts [3]. Each article has events annotated as well as the relations between pairs of events. Section 2 discusses the approach taken to extract finite temporality strings from the texts based on the annotated relations between events. Section 3 gives a probabilistic analysis of the strings, focusing on the likely relations to hold between events that are not themselves directly related in the texts. Section 4 concludes the report as well as giving further improvements that may be done in the future, and is followed by acknowledgements.

## 2 STRING EXTRACTION

Before diving into the explanation of how these strings can be extracted from TimeML annotated documents, it is first important to understand the structure of TimeML, which is an XML Markup Language with several different tags however the main one we focus on here in this report is the `TLINK` tag. Let us take the following example[1] from the TimeBank corpus.

```
<TLINK lid="l14" relType="DURING"
↪  eventInstanceID="ei94"
↪  relatedToTime="t23"/>
<TLINK lid="l1" relType="INCLUDES"
↪  eventInstanceID="ei87"
↪  relatedToTime="t19"/>
<TLINK lid="l2" relType="AFTER"
↪  eventInstanceID="ei89"
↪  relatedToEventInstance="ei90"/>
<TLINK lid="l3" relType="SIMULTANEOUS"
↪  eventInstanceID="ei87"
↪  relatedToEventInstance="ei85"/>
<TLINK lid="l4" relType="AFTER"
↪  eventInstanceID="ei85"
↪  relatedToEventInstance="ei86"
↪  signalID="s16"/>
<TLINK lid="l5" relType="DURING"
↪  eventInstanceID="ei89"
↪  relatedToTime="t20" signalID="s18"/>
<TLINK lid="l6" relType="AFTER"
↪  eventInstanceID="ei92"
↪  relatedToEventInstance="ei93"/>
<TLINK lid="l7" relType="ENDS"
↪  eventInstanceID="ei95"
↪  relatedToTime="t23" signalID="s22"/>
<TLINK lid="l8" relType="BEFORE"
↪  eventInstanceID="ei91"
↪  relatedToTime="t19"/>
```

Here we have nine `TLINK` tags that relate event instances to time intervals. Note that the `eventInstanceID` and `relatedToEventInstance` attributes refer to a mentioning of an event in the text whereas the `relatedToTime` attribute refers to a time interval *e.g* last Tuesday. Important to note also that `relType`[2] gives the name of the Allen relation that relates the two fluents. The rest of the attributes are of no real importance to our goal of extracting strings from the `TLINK` tags.

We can convert these tags into finite temporality strings easily by looking at the `relType` attribute, choosing the corresponding string from Table 1, and replacing the fluents with those attributed to the `TLINK` tag in question. Doing so for the above example yields the following[3] Listing 2

```
1  | |t23|ei94, t23|t23| |
2  | |ei87|t19, ei87|ei87| |
3  | |ei90| |ei89| |
4  | |ei87, ei85| |
5  | |ei86| |ei85| |
6  | |t20|ei89, t20|t20| |
7  | |ei93| |ei92| |
8  | |t23|ei95, t23| |
9  | |ei91| |t19| |
```

In order to be able to combine these nine strings together and to try to build a timeline of these events, some operations on these strings need to be introduced.

### STRING OPERATIONS

The most basic of operations we can perform on strings is the *superposition*. Given two strings, $s$ and $s'$, $s \,\&\, s'$ is defined as the componentwise union of their subsets:

$$\alpha_1 ... \alpha_n \;\&\; \alpha'_1 ... \alpha'_n := (\alpha_1 \cup \alpha'_1) ... (\alpha_n \cup \alpha'_n) \quad (1)$$

For example we have that `|a|b|` & `|c|d|` = `|a,c|b,d|`. One constraint to this operation is that both strings need to be the same length. Block compression and its inverse allows the lengths of strings to change while still maintaining the temporal information encoded within them. Given a string $s$, we define the block compression of the string $bc(s)$ as the following:

$$bc(s) = \begin{cases} s & \text{if } length(s) \leq 1 \\ bc(\alpha s') & \text{if } s = \alpha \alpha s' \\ \alpha bc(\alpha' s') & \text{if } s = \alpha \alpha' s' \text{with } \alpha \neq \alpha' \end{cases} \quad (2)$$

This removes the stutter in strings *e.g* $bc($ `|a|a|a|b|b|` $) =$ `|a|b|`. This function's inverse, $bc(s)^{-1}$ generates the infinite language of strings that are said to be *bc-equivalent* to $s$. We also define the function $pad_k(s)$ as being equivalent to inverse block compression however only returning the finite set of the bc-equivalent strings of $s$ of length $k$.

### ASYNCHRONOUS & VOCABULARY CONSTRAINED SUPERPOSITION

In order to superpose strings that are not the same length, two operations are provided, *asynchronous superposition* and *vocabulary-constrained superposition*. The former is defined as the following in Woods et al. [6]:

$$s \,\&_* \, s' = \{bc(s'') | s'' \in pad_{n+n'-1}(s) \,\&\, pad_{n+n'-1}(s')\} \quad (3)$$

[1] This example is taken from `wsj_0991.tml`

[2] The TimeML standard omits the overlaps relation and also has different names for some of the relations mentioned in Table 1

[3] This was done via the command `$ python3 -m extract wsj_0991.tml -p`

where $n$ and $n'$ are the lengths of $s$ and $s'$ respectively. When superposing 2 strings that represent the temporal relationships between events, the strings in their asynchronous superposition, $s \ \&_* \ s'$ cannot all be equally considered as probable timelines of the events within $s$ and $s'$ as strings in this set can be temporally inconsistent with regards to $s$ and $s'$, or not well formed *i.e* the strings contain intervals that disobey the behaviour of having one and one only beginning and end point. An algorithm described in [6] that narrows this set to those strings that are temporally consistent and well formed is based on only allowing superposition of strings whose matching symbols are located at the same indices. While this approach is correct however it becomes largely unfeasible when trying to extract strings from TimeML documents where events are high in number and the links between them are comparatively low, allowing for a very large number of possible timelines[4]. This approach generates all possible strings first and then narrows down the result which is computationally intensive and slow. A faster and more efficient approach is adopted in *vocabulary constrained superposition* which interleaves generation with testing.

Let $voc(s)$ be defined as the union of all its components. Woods et al. (2018) [5] defines vocabulary constrained superposition as the following:

$$s \ \&_{vc} \ s' = s \ \&_{voc(s),voc(s')} \ s' \tag{4}$$

Given string $s_1 = \alpha s$ and $s_2 = \alpha' s'$ and sets $\Sigma$ and $\Sigma$', then

$$s_1 \ \&_{\Sigma,\Sigma'} \ s_2' = \begin{cases} \{(\alpha \cup \alpha')s'' | s'' \in S\}, \\ \text{if } \Sigma \cap \alpha' \subseteq \alpha \text{ and } \Sigma' \cap \alpha \subseteq \alpha' \\ \varnothing, \text{ otherwise} \end{cases} \tag{5}$$

where

$$S = (\alpha s \ \&_{\Sigma,\Sigma'} \ s') \cup (s \ \&_{\Sigma,\Sigma'} \ \alpha' s') \cup (s \ \&_{\Sigma,\Sigma'} \ s') \tag{6}$$

It is important to note that both asynchronous and vocabulary constrained superposition are commumative and associative so we can superpose any number of strings together regardless of their ordering.

## STRING PARTITIONING

The advantage of vocabulary constrained superposition as opposed to asynchronous superposition is immense in terms of effiency and is described further in [5]. However even this is not enough to mitigate the "combinatorial explosion" that occurs when extracting strings from the TimeML documents in the TimeBank corpus. The documents contain a lot of events with few links between them, much fewer than the $n(n-1)/2$ links needed for $n$ fluents. A solution proposed in [6] is to group strings together that have matching fluents and superpose only those, generating sets of strings with disjoint fluents that represent possible timelines with respect to those fluents within that set of strings. Taking the example where our alphabet $A = \{e_1, e_2, ..., e_{10}\}$ of 10 fluents, we can let the set $G = \{\{e_i\} \mid \forall e_i \in A\}$. Then let $G'$ be the set of sets of fluents that are related by a relation, *i.e* a `TLINK` tag exists that relates them both. In the previous example in Listing 2 we have that

$G = \{\{ei94\}, \{ei87\}, \{ei89\}, \{ei90\}, \{ei85\}, \{t20\}, \{ei93\}, \{ei92\}, \{t23\}, \{ei95\}, \{ei91\}, \{t19\}\}$

and

$G' = \{\{ei94, t23, ei95\}, \{ei87, t19, ei85, ei86\}, \{ei87, t19, ei91\}, \{ei90, ei89, t20\}\{ei93, ei92\}\}$.

Finally for each pair of sets, $g_i', g_j' \in G'$, form the set $g_{ij}' = g_i' \cup g_j'$ iff $\#(g_i' \cap g_j') > 0$ and add it to $G''$, discarding $g_i'$ and $g_j'$. This leaves us with

$G'' = \{\{t23, ei94, ei95\}, \{ei85, ei87, ei86, t19, ei91\}, \{ei90, ei89, t20\}, \{ei93, ei92\}\}$

Now if we refer to each of the clusters of fluents in $G''$ as $C_1, C_2, C_3, C_4$, we can generate sets of strings that correspond to each cluster by taking strings that contain the fluents mentioned in each cluster and grouping them together. Again continuing with our running example taken from the `wsj_0991.tml` document, we get the following clusters of strings:

```
C_1
| |t23|t23, ei94|t23| |,
| |t23|ei95, t23| |

C_2
| |ei85, ei87| |,
| |ei86| |ei85| |,
| |ei87|t19, ei87|ei87| |,
| |ei91| |t19| |

C_3
| |ei90| |ei89| |,
| |t20|ei89, t20|t20| |

C_4
| |ei93| |ei92| |
```

Superposing only strings that belong to the same cluster allows us to generate minimal sets of possible timelines that are easier to compute than trying to superpose every string together. One disadvantage of this approach is that we cannot reason about the temporal relations that may hold between fluents that belong in different clusters. There is nothing limiting us from superposing every

[4]On my laptop that has 4GB of RAM and an Intel i3 processor, the process ate up all memory and would never produce a result for the small example of `wsj_0991.tml`

string together other than computing power. The strings are grouped in this manner to allow for us to circumvent this practical limitation in order to allow us to reason temporally about fluents that are not directly related via a `TLINK` tag but are within the same cluster.

## THE `extract` PYTHON LIBRARY

Accompanying this report is the source code for a python library named `extract`, that generates finite temporality strings from TimeML annotated documents. This library, which can also be used a command-line tool to perform the following functionality[5]:

- Extract corresponding strings from the `TLINK` tags in a given TimeML document.
- Superpose, both asynchronously and via the constrained vocabulary method, any number of strings.
- Group strings using the methodology described in Section 2 and superpose each group individually.
- Output the results to the terminal or store them into a file.

Every previous string operation mentioned in Section 2 is implemented, as well as those introduced later, allowing users of the library to manipulate any finite temporality string and are not just constrained to using it to extract strings from TimeML annotated documents.

Taking the cluster $C_1$ from the previous section, we get a the following list[6] of five possible timelines of events when we superpose its strings together:

```
| |t23|ei95, t23|ei95, t23, ei94|ei95, t23| |
| |t23|t23, ei94|ei95, t23, ei94|ei95, t23| |
| |t23|t23, ei94|t23|ei95, t23| |
| |t23|t23, ei94|ei95, t23| |
| |t23|ei95, t23, ei94|ei95, t23| |
```

# 3 ANALYSIS

> Given an Allen relation $R$, what is the probability that $R$ relates intervals $a$ and $a'$, $aRa'$?

This is the underlying motivation for the work by Fernando et. al. [2], starting from first principles until probabilities about relations are calculated. Here we try to answer this question through the frame of the TimeBank corpus.

[5]For more information on how to use this library, consult its documention, or email its author at `sulimanm@tcd.ie`.

[6]This was done via the command `$ python3 -m extract wsj_0991.tml -g`

## THE TIMEBANK CORPUS

Before beginning an analysis of the strings generated, it is important to include a discussion about the accuracy and correctness of the annotations in the TimeBank corpus. Manual annotation is often subject to inaccuracies, be it from annotator fatigue or subjectivity among different annotators. A good measure of the accuracy of annotations is the *inter-annotator agreement*, which quantifies how much annotators differ in their annotations of the same document. Values can range from 0 to 1 with higher values intrepeted as meaning a high level of agreement between annotators. The TimeBank Corpus [4] reports an inter annotator agreement value for the `EVENT` tag, the tag that categorizes a specific event as 0.78, and 0.83 for `TIMEX3` tags, those that categorize time intervals. These values are high enough for us to assume that the categorization of events *i.e* the fluents, is correct and that none have been omitted or incorrectly labelled for the purposes of this analysis.

However when it comes to indentifying the links between events and time intervals, an inter annotator agreement score of 0.55 is given for the `TLINK` tags corpus. This indicates a high level of disagreement as to how relations between events should be categorized.

## RELATING THE UNRELATED

For each file in the TimeBank corpus, we can cluster the strings via the way described in Section 2 and superpose strings in each cluster together[7]. For each file, let $S_i$ be the set of strings *i.e* possible timelines, resulting from the superposition of strings in the $i$-th cluster, $C_i$. $S_i$ is a language with alphabet $A_i$ and $\forall s \in S_i,\ voc(s) = A_i$. We know this because all the strings in cluster $C_i$ have been superposed together resulting only in strings that contain every fluent mentioned in the strings in cluster $C_i$.

Given any string $s \in S_i$, and any pair of fluents $a, a' \in A_i$ in $s$, we can determine their temporal relation in the string $s$ by the following operation:

$$bc(\rho_{\{a,a'\}}(s)) \tag{7}$$

where $s = \alpha_1 ... \alpha_n$ and

$$\rho_X(s) = (\alpha_1 \cap X)\ ...\ (\alpha_n \cap X) \tag{8}$$

Let us view an example with $s =$ `| |t23|t23, ei94|ei95, t23| |`:

$bc(\rho_{\{ei94,ei95\}}(s)) =$ `| |ei94|ei95| |`

[7]Note that I was able to accomplish this on 100 of the 183 articles. Even with clustering, some of the groups of strings were too large for my computer to be able to compute the timelines.

If we do this for every string in Listing 2 we can see how fluents `ei94` and `ei95` relate to each other in the 5 possible timelines, even though in the TimeML document there is no `TLINK` tag relating these two fluents.

```
| |ei95|ei95, ei94|ei95| | ei94 during   ei95
| |ei94|ei95, ei94|ei95| | ei94 overlaps ei95
| |ei94| |ei95| |          ei94 before   ei95
| |ei94|ei95| |          ei94 meets    ei95
| |ei95, ei94|ei95| |     ei94 starts   ei95
```

Each possible relation between `ei94` and `ei95` is shown above, given the context of the strings in their cluster. Here we can see that in 3 out of the 5 timelines, `ei94` starts before `ei95`, and in all of the timelines, `ei95` ends after `ei94` has passed. Even though they are not both directly related, we are able to reason about them.

For every pair of unrelated fluents $a$ and $a' \in A_i$ of a given language $S_i$, $bc(\rho_{\{a,a'\}}(s))\ \forall s \in S_i$, gives how they may be related in each timeline $s$. We can count the amount of times each pair of fluents are related via a given relation, and aggregate the counts together over every cluster in every file in the corpus. The raw counts for each relation are given in Figure 1. Note that only 8 of the 13 relations have non zero counts. This is because we do not consider both of the pairs $(a, a')$ and $(a', a)$ as the relation $R$ that holds $a'Ra$ is just the inverse of the relation $R'$ that holds $aR'a'$ and does not provide any extra useful information.

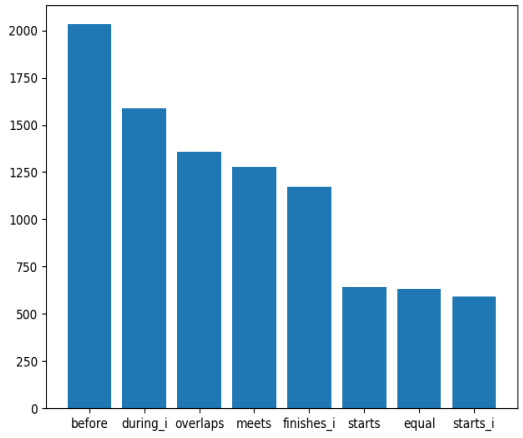


Figure 1: The raw counts of relations for unrelated fluents

## DISCUSSION

In total, just under 10000 pairs[8] of unrelated intervals were analysed to see how they may possibly relate to each other. We can get the proportion of pairs of fluents that are related by a

[8]The actual value is 9296 pairs.

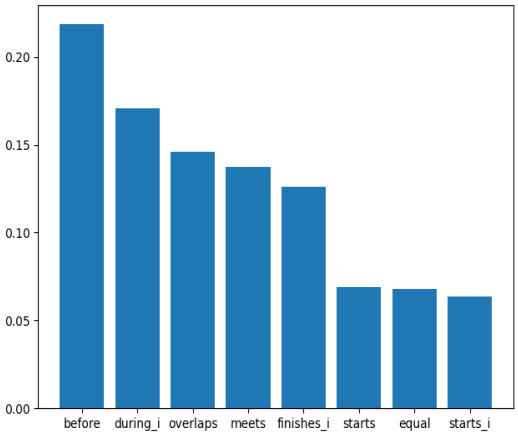


Figure 2: The proportion of relations between unrelated fluents for every file

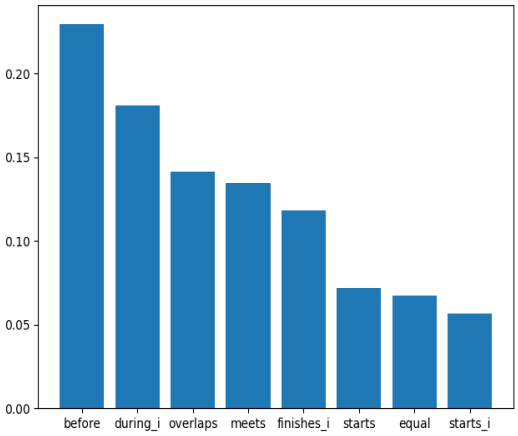


Figure 3: Political reporting proportions

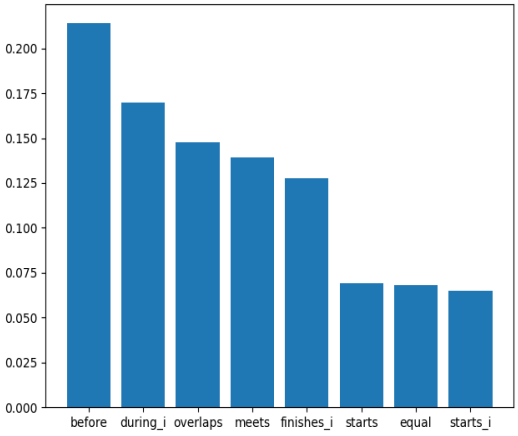


Figure 4: Corporate reporting proportions

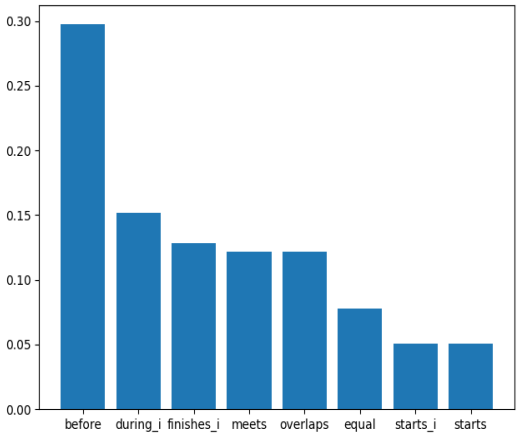


Figure 5: Crime reporting proportions

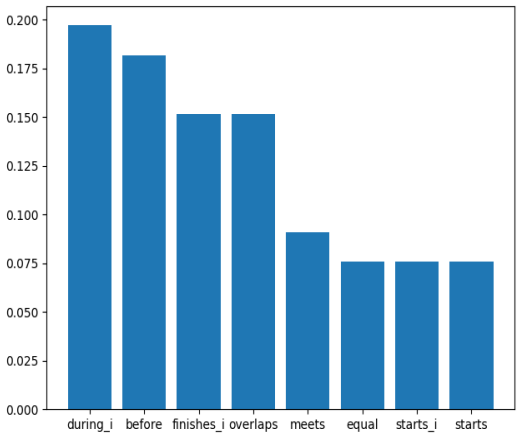


Figure 6: Science reporting proportions

given Allen relation by dividing the count over the total number of pairs. This is shown in Figure 2. Here we see that slightly over 20% of the time when we have two unrelated fluents, they are related via the before relation. Now let us consider one of these relations: meets.

Given two unrelated fluents $a$ and $a'$ what is the probability that they are related via the meets relation?

This can be rephrased as $P(R = m|aRa')$. If we have our population of $n = 9296$ pairs of unrelated fluents and from Figure 2 we see that the proportion of pairs in the population that are related via meets is $\pi = 0.137$, it follows then the *binomial(n, $\pi$)* probability distribution.

Splitting the files into political reporting, crime reporting, corporate reporting, and scientific reporting, we get the proportions of relations shown in Figures 3 - 6. Each category of news reporting tends to follow generally the same pattern of relations, with crime reporting having a much larger proportion of unrelated events being related by the before relation, around 0.30. These proportions speak to the style of writing in newspapers, and the large proportion of before relations being used in crime reporting could be attributed to the fact that the crimes being reported have already happened.

# 4 CONCLUSION

In this report we have explored the idea of using finite temporality strings to represent temporal information extracted from TimeML documents. We have superposed strings together to try and build a picture in time of how events that occur in these documents relate to one another, and included an analysis of how those events that have no direct relation may be related given the context.

Further improvements to be made include the ability to superpose every string together in a give TimeML document, and not having to cluster them. This would then allow us to reason between any pair of fluents, not just those that belong in the same cluster.

# 5 ACKNOWLEDGEMENTS

I would like to thank Carl Vogel for his input and assistance, and I am grateful for his continued guidance throughout this project. I would also like to thank David Woods for his help with the algorithms described here.

# References


[1] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, 1983.

[2] Tim Fernando and Carl Vogel. Prior probabilities of allen interval relations over finite orders. In *Proceedings of the 11th International Conference on Agents and Artificial Intelligence*, page nil, - 2019.

[3] James Pustejovsky. *ISO-TimeML and the Annotation of Temporal Information*, pages 941–968. Handbook of Linguistic Annotation. Springer Netherlands, 2017.

[4] James Pustejovsky, Patrick Hanks, Roser Saurí, Andrew See, Rob Gaizauskas, Andrea Setzer, Dragomir Radev, Beth Sundheim, David Day, Lisa Ferro, and Marcia Lazo. The timebank corpus. *Proceedings of Corpus Linguistics*, 01 2003.

[5] David Woods and Tim Fernando. Improving String Processing for Temporal Relations. page 11.

[6] David Woods, Tim Fernando, and Carl Vogel. Towards Efficient String Processing of Annotated Events. page 10.